Interactive Verification of Medical Guidelines

Jonathan Schmitt[1], Alwin Hoffmann[1], Michael Balser[1],
Wolfgang Reif[1], and Mar Marcos[2]

[1] Lehrstuhl für Softwaretechnik und Programmiersprachen
Universität Augsburg, D-86135 Augsburg, Germany
{schmitt, balser, reif}@informatik.uni-augsburg.de
[2] Dept. of Computer Engineering and Science, Universitat Jaume I
Campus de Riu Sec, E-12071 Castellón, Spain
Mar.Marcos@icc.uji.es

Abstract. In the medical domain, there is a tendency to standardize health care by providing medical guidelines as summary of the best evidence concerning a particular topic. Based on the assumption that guidelines are similar to software, we try to carry over techniques from software engineering to guideline development. In this paper, we show how to apply formal methods, namely interactive verification to improve the quality of guidelines. As an example, we have worked on a guideline from the American Academy of Pediatrics for the management of jaundice in newborns. Contributions of this paper are as follows: (I) a formalized model of a nontrivial example guideline, (II) an approach to verify properties of medical guidelines interactively, and (III) verification of a first example property.[1]

1 Introduction

There is a tendency in the medical domain to standardize health care. This is because the amount of medical studies conducted per year has long passed a point, where every doctor can keep track of all results and also judge their quality. To ease the workload, the instrument of guidelines has been devised. Guidelines represent a summary of the best evidence concerning the interventions to manage a particular clinical condition. Guideline developers take over the cumbersome work of literature search and the evaluation of the quality of the relevant studies. All this data is then compiled into a document, usually of 50 to 150 pages, where medical staff can access the relevant recommendations in a fast, efficient way. This makes guidelines very important documents, as hundreds to thousands of physicians may act upon them, treating several thousands of patients according to these guidelines. An error within such a guideline may cause great harm and therefore, the quality of the guideline itself must be assured.

A first step to improve the quality of guidelines has been to introduce the AGREE instrument[2], which is an informal review, mainly concerned with the

[1] This work has been partially funded by the European Commission's IST program, under contract number IST-FP6-508794 Protocure II.

[2] http://www.agreecollaboration.org

guideline development process. However, the quality of the content is less in the focus. Guidelines are written as natural language text and may therefore contain ambiguities in the formulation, ambiguities in the description of the treatment process or even wrong information, wrongly transferred into the document. The quality of the content of guidelines is the scope of a European research project called Protocure[3].

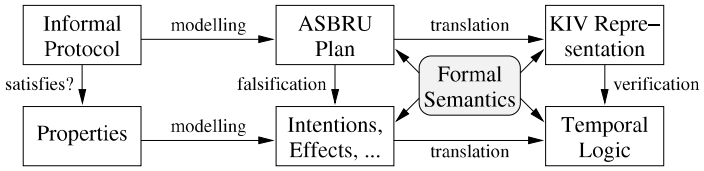

Fig. 1. Formalization and verification of medical guidelines

The Protocure project aims to assist the guideline development process by providing tools and techniques to evaluate not the process but the contents of a guideline during its complex development process. Based on the assumption that guidelines are similar to software, we try to carry over techniques from software engineering to guideline development. Our approach is to formalize guidelines and to verify properties (see Fig. 1). Guidelines are modeled in Asbru [1,2], which is a planning language especially designed for the medical domain. The modeling process already improves the quality of the guideline [3]. With formal verification, more errors can be revealed. For this, we have defined a formal semantics of Asbru in [4] and have translated Asbru to the SMV model checker [5]. In this paper, we take a different approach and focus on interactive verification of Asbru. As an example, we have worked on a guideline from the American Academy of Pediatrics for the management of jaundice in newborns.

Contributions of this paper are as follows: (I) a formal Asbru model of a non-trivial example guideline, (II) an approach to verify properties of medical guidelines interactively, and (III) verification of a first example property. In Sect. 2, we give an overview of the Asbru language, Sect. 3 introduces the medical guideline. Support for Asbru in the interactive theorem prover KIV [6] is described in Sect. 4, and the example property is verified in Sect. 5. Section 6 gives related work and Sect. 7 concludes.

2 Introduction to Asbru

Asbru is a hierarchical planning language. Basis of the Asbru semantics is the concept of plans with a defined plan state model visualized in Figure 2. All transitions within this state-chart are guarded, where the guards `SC`, `RA` and `E` represent external signals, which are sent to the plan by its respective super-plan. Evaluation of the other guards are dependent on the so called Asbru conditions. These are mappings from medical relevant data to boolean values. Details of the mappings are dependent on the individual Asbru plan.

[3] http://www.protocure.org

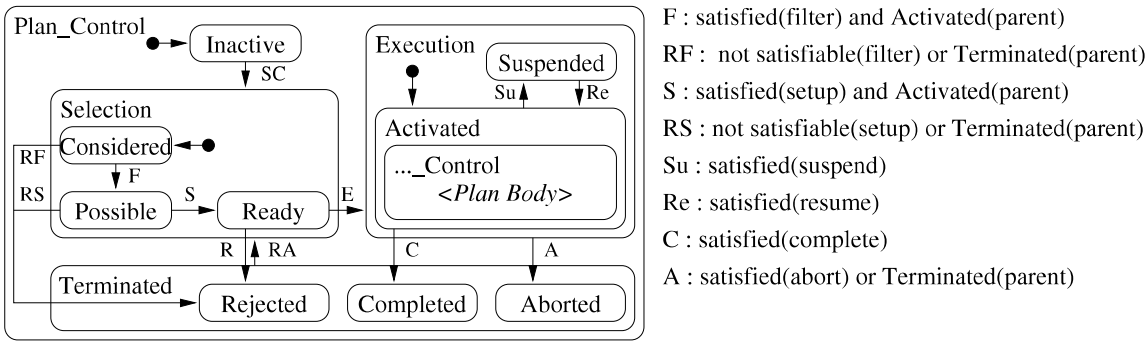

Fig. 2. Asbru semantics

Behavior of Asbru plans during their execution phase is dependent on their plan type. A plan can be of type `sequential`, meaning that the plan will start all of its sub-plans sequentially in the given order. If the plan type is `anyorder`, plans are still executed sequentially but the order of the execution of the sub-plans is arbitrary. It is also possible to start multiple sub-plans at a time. For this, Asbru provides the `parallel` and `unordered` plan types, where parallel-arranged sub-plans are synchronized, the unordered-controlled sub-plans are not. Furthermore, special plan types are defined for the request of medical data, calculations or deterministic choices.

2.1 State

For a graphical representation of an abstracted view of the data flow, see Figure 3. An Asbru system, consisting of Asbru plans and an environment, can be seen as a plan based implementation of a medical decision support system. The system bases its decisions upon data being provided by medical staff. This data is provided in a data structure called Patient Data • • for the system. In the Asbru implementation, the provider of medical data – the medical staff – is called the `monitoring unit`. Data may be abstracted from numerical values to symbolic values. As all data types in Asbru are discrete, data can be transformed to the smallest occuring unit in the case study, thus eliminating the need to annotate physical units.

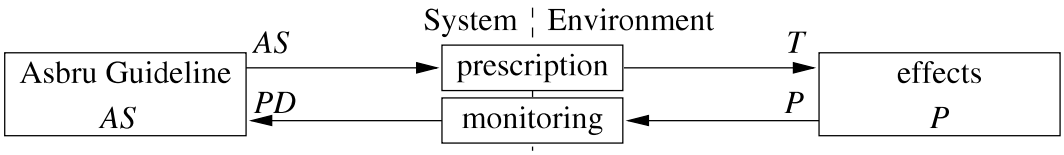

Fig. 3. Abstracted data flow within Asbru

The monitored data is processed according to the guideline represented as Asbru plans, and the output of this processing is provided to the medical staff as an abstracted description of the states of several Asbru plans, which are combined into the variable Asbru state • • . Medical staff, receiving this data, interprets it to come up with a set of treatments • to be administered to the patient • .

The condition of the patient changes in the environment. This transformation is described by `effects`. Treatments applied to the patient may restrict the otherwise indeterministic patient behavior. Effects are applied to the patients condition and the result of this application is then handed over to the monitoring unit. The monitoring unit compiles this data according to the decision of the medical staff. Therefore, there may be a discrepancy between the real condition of the patient and the known subset of it.

2.2 Plan State Model

The plan state model is depicted in Figure 2. A detailed description of a Structured Operational Semantics is given in [4]. For example, a plan, currently `considered` has the options of either advancing its state to `possible` or to change its state to `rejected`. This choice is dependent on the guards `F` and `RF`, where the `F` guard is satisfied, if and only if the filter condition of the plan is satisfied and the super-plan is still active. The `RF` guard is satisfied, if the super-plan is terminated or the filter condition is no longer satisfiable.

Six conditions define the transitions of an Asbru plan through the plan state model: `filter`, `setup`, `suspend`, `reactivate`, `abort` and `complete` conditions. Termination and suspension behavior is also affected by the states of sub- and super-plans.

2.3 Time-Annotated Conditions

Conditions are first order formulas φ mapping a state to a truth value. Conditions can be time-annotated. A time-annotated condition is written as follows

$$\varphi\ [\bullet\bullet\bullet, \blacksquare\bullet\bullet][\bullet\bullet\bullet, \blacksquare\blacksquare\bullet][\bullet\ \blacksquare\bullet\bullet\ \bullet, \bullet\ \bullet\bullet\bullet\ \bullet]\ \bullet\bullet\bullet$$

where $\bullet\bullet\bullet$ is the earliest starting shift, $\blacksquare\bullet\bullet$ the latest starting shift, $\bullet\bullet\bullet$ the earliest finishing shift, $\blacksquare\blacksquare\bullet$ the latest finishing shift, and $[\bullet\ \blacksquare\bullet\bullet\ \bullet, \bullet\ \bullet\bullet\bullet\ \bullet]$ the duration interval. The starting interval is defined as $[\bullet\bullet\bullet + \bullet\bullet\bullet, \blacksquare\bullet\bullet + \bullet\bullet\bullet]$. The finishing interval is defined accordingly. Any value but the reference point may be omitted and is then written as '_'. As a reference point, an absolute time point can be given as well as enter($\bullet\blacksquare\bullet\bullet\blacksquare\bullet\bullet\bullet$, $\bullet\bullet\bullet\ \bullet$) and leave($\bullet\blacksquare\bullet\bullet\blacksquare\bullet\bullet\bullet$, $\bullet\bullet\bullet\ \bullet$) to refer to the time when plan $\bullet\bullet\bullet\ \bullet$ enters or leaves state $\bullet\blacksquare\bullet\bullet\blacksquare\bullet\bullet\bullet$. Also, with `*now*`, we refer to the time of evaluation.

Time annotations represent sets of intervals ranging over time. An interval is member of a time annotation if and only if its starting point is member of the starting interval, its finishing point is member of the finishing interval and its duration is member of the duration interval. A time-annotated condition evaluates to true, if and only if there is an interval $\bullet$ which is element of the time annotation and for which the condition is true at all times. Furthermore, if an earliest starting shift or a maximum duration is given, the condition must be initially false, more precise, in the time point prior to the starting point of $\bullet$ the condition must evaluate to false. Similarly, if a latest starting shift or a maximum duration is given, the condition must finally evaluate to false.

2.4 Intention

Asbru intentions model the aims of a plan. They are formulated in terms of temporal patterns making reference to plans and/or state variables that should be maintained, achieved or avoided, during or after the execution of a particular plan. In this sense, they can be regarded as proof obligations for Asbru. Intentions are written down similar to conditions. Intentions can also be time-annotated.

3 Example Medical Guideline

3.1 Jaundice Guideline

Jaundice (or hyperbilirubinemia) is a common disease in newborn babies. Under certain circumstances, elevated bilirubin levels may have detrimental neurological effects. In many cases jaundice disappears without treatment but sometimes needs phototherapy to reduce the levels of total serum bilirubin (TSB), which indicates the presence and severity of jaundice. In a few cases, however, jaundice is a sign of a severe disease.

The jaundice guideline of the American Association of Pediatrics (AAP) [7] is intended for the management of the disease in healthy term newborn babies. The main reason for choosing this guideline was that it is considered a high-quality one: as a matter of fact, the AAP jaundice guideline has been included in the repository of the National Guideline Clearinghouse[4] until it was replaced by a more recent update.

This particular guideline is a 10 pages document. It consists of an evaluation (or diagnosis) part and a treatment part, to be performed in sequence. During the application of the guideline, as soon as the possibility of a more serious disease is uncovered, the recommendation is to exit without any further action. The rationale behind this is that the guideline is exclusively intended for the management of jaundice in otherwise healthy newborns.

3.2 Jaundice Guideline in Asbru

Like the AAP guideline, the Asbru version has as main components a diagnostics part and a treatment part. It is made up of about 40 plans and has a length of 16 pages in an intermediate Asbru notation. Figure 4 shows part of the Asbru model for the jaundice guideline with an emphasis on treatment. The links represent the decomposition of plans into steps or sub-plans, and the plan type is represented by different kinds of arrows. Besides, the waiting strategy is indicated either with bold font keywords (e.g. **one** or **all**) or with bold font plan names (e.g. the "Observation" plan is a compulsory part of the plan group marked "*" in Figure 4).

The "Check-for-..." plans model monitoring of TSB level and check-ups at specific time intervals. The most important entry point of the guideline is the plan "Diagnostics-and-treatment-hyperbilirubinemia". It is divided into a diagnostics sub-plan and a treatment one, to be executed sequentially. Next we will focus on the latter.

[4] `http://www.guideline.gov/`

The treatment phase consists of two parallel parts, namely the actual treatments and a cyclical plan asking for the input of TSB values and updating the infants age every 12 to 24 hours. Regarding the treatments (label (**) in Fig. 4), either the regular ones ("Regular-treatments") or a transfusion ("Exchange-transfusion") can take place depending on the bilirubin level.

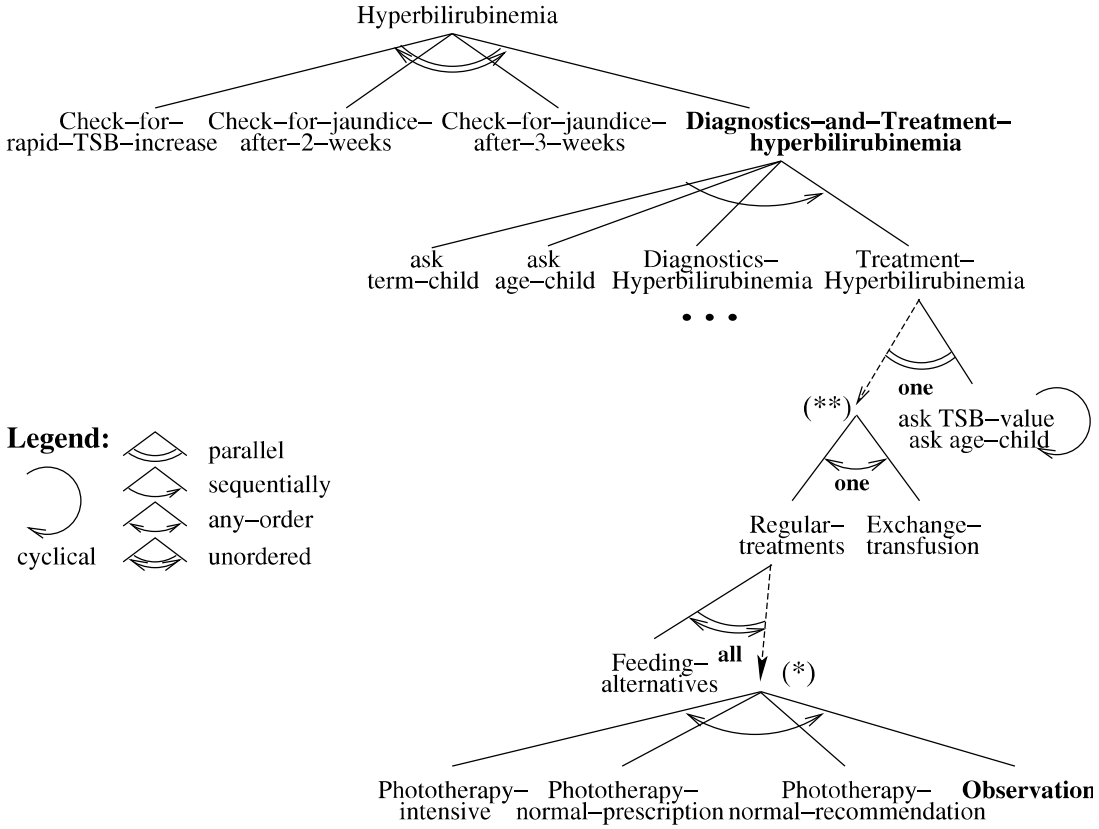

Fig. 4. Overview of the jaundice model in Asbru, with details of the treatment part

The "Regular-treatments" plan contains the proper treatment procedures. It consists of two parts to be performed in parallel (unordered): the study of feeding alternatives and the different therapies (see label (*)). The plans in group (*) can be tried in any order, one at a time. The intentions of "Regular-treatments" plan are both avoiding toxic bilirubin levels and attaining normal (or observation) ones at the end. The plan completes when the feeding alternatives and the therapies complete (waiting strategy **all**). The latter in turn depends on the completion of observation (compulsory plan in bold).

3.3 Plan "Phototherapy-Intensive"

As an illustration, we describe the plan "Phototherapy-intensive" together with its conditions and intentions. The intermediate Asbru notation for the plan is shown below:

plan Phototherapy-intensive
intentions
maintain intermediate-state:
(and (TSB-decrease = yes) *in* ([4h, -] [-, 6h] [-, -] SELF)
(TSB-change ≥ 1) *in* ([4h, -] [-, 6h] [-, -] SELF)**)**
conditions
filter-precondition:
(or (bilirubin = phototherapy-intensive) *in* NOW
`normal-phototherapy-failure`**)**
abort-condition:
(or (and (bilirubin ≠ phototherapy-intensive) *in* NOW
(not `normal-phototherapy-failure`**))**
`intensive-phototherapy-failure:`
(and (bilirubin = phototherapy-intensive) *in* NOW
(or (and (TSB-decrease = yes) *in* ([4h, -] [-, 6h] [-, -] SELF)
(TSB-change < 1) *in* ([4h, -] [-, 6h] [-, -] SELF)**)**
(TSB-decrease = no) *in* ([4h, -] [-, -] [-, -] SELF)**)**
) *explanation* "Failure of intensive phototherapy."
)
plan-body
Prescribe-intensive-phototherapy

The plan body simply contains an activation of a user-performed plan representing the clinician's action, "Prescribe-intensive-phototherapy". The most important elements of the above plan are its conditions and intentions. Among the former we can distinguish the filter preconditions specifying the eligibility criteria for the plan, and the abort conditions describing the situations in which it should be interrupted. The label `normal-phototherapy-failure` is defined elsewhere as **(and** (bilirubin = phototherapy-normal) *in* NOW (TSB-decrease = no) *in* NOW**)**. The abort conditions include not only the situation in which the bilirubin levels change to a value not requiring intensive phototherapy (different from phototherapy-intensive), but also the case of therapy failure – according to the guideline, [illegible] and [illegible].

An intention was specified by the modelers for the plan "Phototherapy-intensive". Since the guideline states that intensive phototherapy should produce a decline in the bilirubin levels of 1 to 2 mg/dl in 4 to 6 hours, it has been considered that this therapy aims at lowering the TSB levels in such a degree within this time range. This intention is intimately related to part of the abort conditions of the plan. It is specified to abort if there is an insufficient or no decrease in the TSB levels in the specified time range. In our verification case-study we have analyzed this intention.

4 Integration of Asbru in KIV

KIV is an integrated development environment to develop systems using formal methods. Systems are specified with algebraic specifications. System properties

are verified by constructing proofs in an interactive theorem prover which is based on higher order logic with special support for temporal logic. KIV has been very useful for the development of sequential programs. Support for the development of concurrent systems and especially Asbru has recently been added.

4.1 Temporal Logic in KIV

KIV offers support for future-time linear temporal logic. Reactive systems can be described in KIV by means of state-charts or parallel programs; here we use parallel programs. A state of a system is encoded in first-order logic. Static variables $\bullet$, which have the same values at each time point, are distinguished from dynamic variables $\bullet$. A specialty of KIV is the use of primed and double-primed variables [8]: a primed variable $\bullet\,'$ represents the value of this variable after a system transition, the double-primed variable $\bullet\,''$ is interpreted as the value after an environment transition. System and environment transitions alternate, with $\bullet\,''$ being equal to $\bullet$ in the successive state. Here, an Asbru guidelines defines the system transition and the patient is interpreted as the environment. The supported future-time temporal operators include $\Box\,\varphi$ ("always φ"), $\Diamond\,\varphi$ ("eventually φ) and others.

4.2 Specification of Asbru

We have modeled the Asbru language in the interactive theorem prover KIV as follows. The data types are defined with algebraic specifications, the dynamic behavior, i.e. the plan state model of Asbru, is encoded in a parallel program. An overview of the algebraic specifications is given in Figure 5.

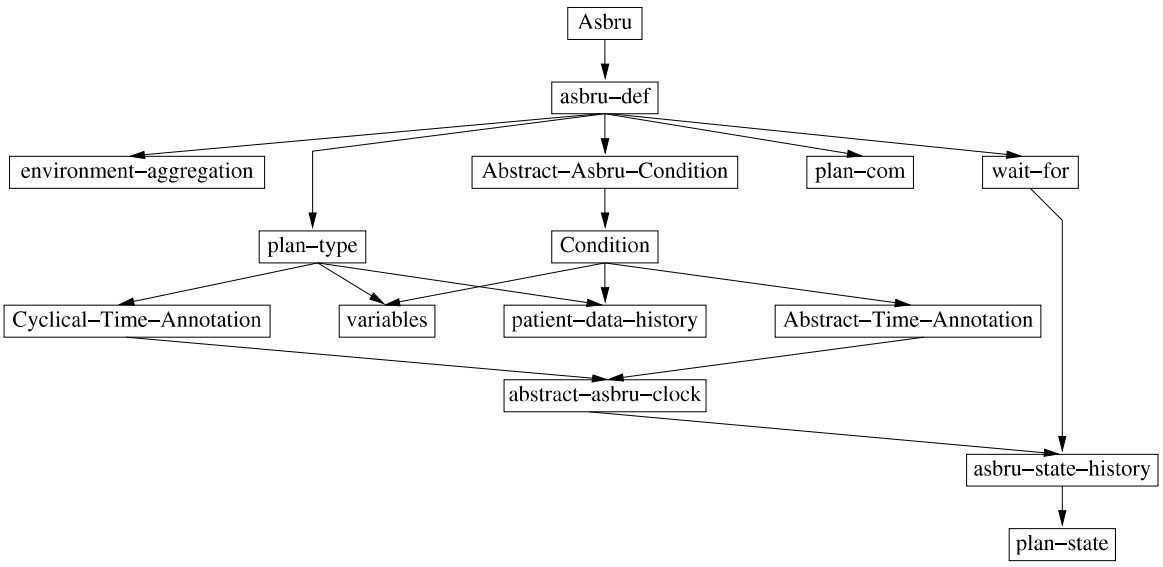

Fig. 5. Algebraic specifications defining the Asbru language

Asbru is based on a discrete micro/macro step semantics. Micro steps are internal plan transitions. If the plan state is stable a macro step is executed and

the systems interacts with the environment. A discrete Asbru clock • • with micro and macro step counter is defined in •••••••••••••••••••••. The definition of time annotations in • ••••••••• •• ••• •••••••••• is based on Asbru clocks. The specification defines a tuple containing the starting and finishing shifts, the duration and the reference point. The tuple represents a set of intervals. Conditions are modeled as higher order predicates which are evaluated for a given state. Non-trivial predicates satisfied and satisfiable in specification • •••••••• define the semantics of time annotated conditions. • ••••••••• •••••• •••••••• extends normal conditions by additional signals, necessary for advanced Asbru features which are omitted here.

The patient data • • stores knowledge about the state of the patient. A generic container is defined in •••••••••••••••••••• and can be instantiated for the concrete case study. The patient data history • • • maps time to patient data. This history is important to evaluate time annotated conditions. An enumeration type •••••••••• defines the different plan states `inactive`, `considered`, etc. An Asbru state • • maps plan names to plan states. A history of Asbru states • •• is defined in •••••••••••••••••••. The different plan types such as `sequential`, `anyorder`, `unordered`, and `parallel`are defined in •••••••••. Description of further features such as the wait-for construct defining optional and mandatory sub plans (specification • •••••••), and cyclical plan types (specification • •••••••• •• ••• ••••••••••) are omitted here. Specifications •••••••• •••• •••••••••••• and ••••••••• accumulate the environment signals.

```
considered#(sk, sk0; var AC, PDH, ASH, AS, ...)
begin
  if plan-not-filter(sk, AC, PDH, ASH, AS, ...) then
    AS[sk] := rejected
    else if plan-filter(sk, AC, PDH, ASH, AS, ...) then
      AS[sk] := possible;
      possible#(sk, sk0; AC, PDH, ASH, AS, ...)
    else
      AS[sk] := considered;
      considered#(sk, sk0; AC, PDH, ASH, AS, ...)
  end
```

Fig. 6. Dynamic behavior of Asbru

The dynamic behavior is encoded as a parallel program. The different Asbru plans of a plan hierarchy are executed synchronously. A single plan follows the plan state model of Figure 2. The procedure considered# of Fig. 6 implements the transitions originating from state `considered`. Similar procedures inactive#, possible# exist for the other transitions. With this close match between plan state model and implementation, errors in the latter can be spotted easily.

4.3 Translation of Intentions

Intentions are translated to a temporal formula with two boolean variables • • • (Start Event Trigger) and • • • (End Event Trigger). Initially, the variables are

set to false. Once the interval is entered, during which the intention must be satisfied, $\bullet\bullet\bullet$ is set to true. $\bullet\bullet\bullet$ signals the end of the interval. An intermediate-state maintain intention can be translated as follows [9]:

$$\Box\ (\bullet\bullet\bullet\ \wedge \neg\ \bullet\bullet\bullet\ \rightarrow\ \ \bullet\bullet\bullet\ [\bullet\bullet\ ][\texttt{'TSB-decrease'}].\text{val} \\ \wedge\ \bullet\bullet\bullet\ [\bullet\bullet\ ][\texttt{'TSB-change'}].\text{val} > 1)$$

In our verification example, we need to reference the time point, where plan "Phototherapy-intensive" (abb. `'pti'`) gets activated, which can be assessed with the predicate time-enter-state(...). Adding four or six hours to that time point gives us the times, where $\bullet\bullet\bullet$ and $\bullet\bullet\bullet$ have to be set. This setting is done by a TL formula, describing the environment transition of $\bullet\bullet\bullet$ and $\bullet\bullet\bullet$. The desired behavior during the now defined timing period is a constant high decrease of bilirubin in the blood. For increased readability the predicates SET-pred and EET-pred can be used, where

$$\begin{array}{l} \text{SET-pred}(\bullet\bullet\bullet, \bullet\bullet, \bullet\bullet\bullet) \\ \leftrightarrow \textbf{if}\ \text{time-enter-state}(\bullet\bullet\bullet, \texttt{'pti'}, \texttt{activated}, \bullet\bullet) + 4 = \bullet\bullet \\ \quad \textbf{then}\ \bullet\bullet\bullet\ '' \ \textbf{else}\ \bullet\bullet\bullet\ '' \leftrightarrow \bullet\bullet\bullet\ ' \end{array}$$

and EET-pred is defined analogously.

5 Interactive Verification of Asbru

Formal verification of the indicator for the jaundice guideline is rather complex, because of the nontrivial semantics of time annotations. In Section 5.1, we define the proof obligation which is to be verified and in the following sections give an overview of our proof and the errors which were discovered.

5.1 Proof Obligation

The proof obligation falls into four different parts. The current system configuration, the system description, the environment assumption, and the property to verify. Plan Phototherapy-intensive (which is abbreviated with `'pti'`) is initially `inactive`, and the consider signal has been sent. Procedure inactive# defines the dynamic behavior of the Asbru plan in state `inactive` (see Sect. 4.2).

In the $\bullet\bullet\bullet\blacksquare\bullet\bullet\bullet$ $\bullet\bullet\bullet$ $\bullet\bullet\bullet\bullet\bullet$ $\bullet\blacksquare\bullet\bullet$, we ensure that the environment leaves the local system variable $\bullet\bullet$ [`'pti'`] unchanged. Furthermore, either a micro step is executed and the patient data history $\bullet\bullet\bullet$ does not change, or a macro step is executed and the patient state changes arbitrarily. A macro step is only possible, if the system is stable, i.e. variable $\bullet\bullet\bullet'$ is true. The property is already described in Section 4.3.

5.2 Proof Structure

In principle, verification of concurrent systems in KIV is based on symbolic execution with induction [8,10]. The Asbru guideline is executed step by step.

$$
\begin{array}{l}
\quad /^{*}\ \textit{current system configuration}\ ^{*}/ \\
\quad AS\,[\texttt{'pti'}] = \texttt{inactive}, PC\,[\texttt{'pti'}].\text{consider}, \neg\ SET, \neg\ EET, \\
\quad \text{time-enter-state}(ASH, \texttt{'pti'}, \texttt{activated}, AC\,) + \text{num}(4) < AC, \ldots \\
\quad /^{*}\ \textit{system description}\ ^{*}/ \\
\quad [\text{inactive}\#(\texttt{'pti'}, \texttt{'rt'}; AC, PDH, ASH, AS, \ldots)], \\
\quad /^{*}\ \textit{environment assumption}\ ^{*}/ \\
\quad \Box\ (\quad AS''[\texttt{'pti'}] = AS'[\texttt{'pti'}] \wedge\ \ldots \\
\qquad \wedge\ (\quad \text{micro}(AC'', AC') \wedge PDH''[AC''] = PDH\,[AC\,] \\
\qquad\quad \vee\ Tick' \wedge \text{macro}(AC'', AC'))) \\
\quad /^{*}\ \textit{definition of flags SET and EET}\ ^{*}/ \\
\quad \Box\ (\text{SET-pred}(ASH, AC, SET) \wedge \text{EET-pred}(ASH, AC, EET)), \\
\vdash\ /^{*}\ \textit{property}\ ^{*}/ \\
\quad \Box\ (SET \wedge \neg\ EET \rightarrow \quad PDH\,[AC\,][\texttt{'TSB-decrease'}].\text{val} \\
\qquad\qquad\qquad\qquad \wedge\ PDH\,[AC\,][\texttt{'TSB-change'}].\text{val} > 1)
\end{array}
$$

Fig. 7. Sequent to verify

With the verification of an always property, after performing a TL step, it is necessary to verify a FOL proof obligation and afterwards continue TL execution. Symbolic execution may not terminate, because the guideline may contain loops. For executing loops, induction is necessary.

In order to verify the intention, a proof for the correctness of the sequent of Fig. 7 must be constructed. The execution, in principle, follows the abstracted proof tree of Figure 8. The proof tree presented is drawn bottom up. The lowest node in the proof tree is the initial node, marked "1" in Figure 8.

In the initial state, the plan "Phototherapy-intensive" (abb. `'pti'`) is in `inactive` state, as is the sub-plan "Prescribe-intensive-phototherapy" (abb.

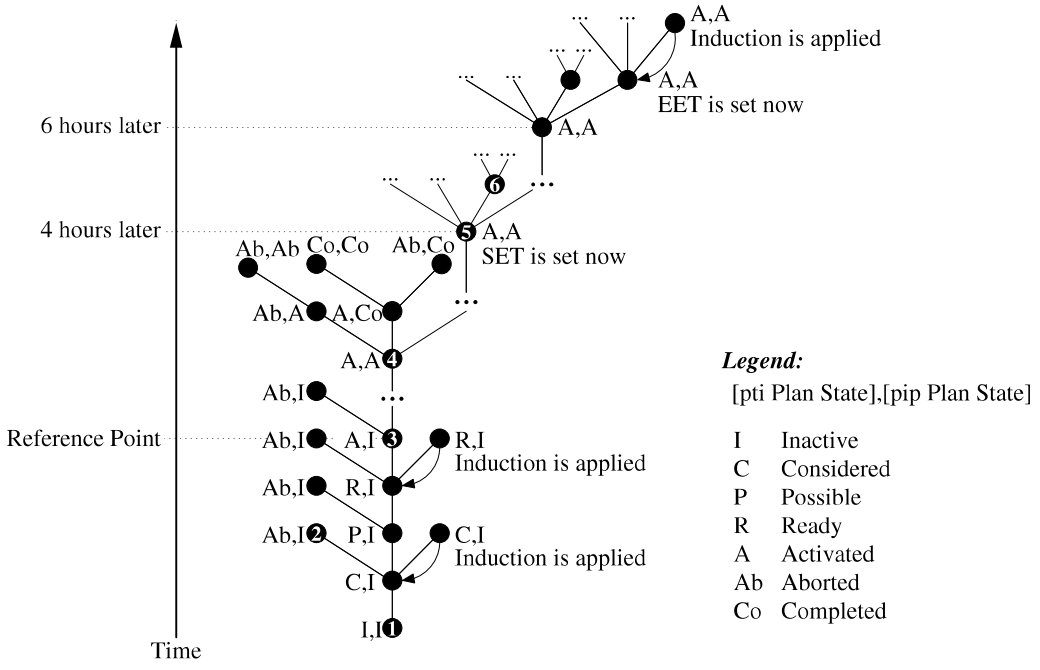

Fig. 8. Abstracted proof tree

'pip'). This is annotated in Fig. 8 as I, I. After advancing one TL step, the plan is `considered`. After each TL step, the possibility has to be considered, that the super-plan of 'pti' is terminated. This will result in the 'pti' plan to terminate as well. An example can be seen with node 2 in Figure 8. Status of the plans there is abbreviated to Ab, I.

We continue to progress until plan 'pti' reaches the `activated` state (see node 3 in Fig. 8). The transition of 'pti' from `ready` to `activated` marks a reference time. The function time-enter-state(ASH, 'pti', activated, AC) is used to determine that time. This is necessary, as this reference time is used as a reference point for time annotations in the abort condition as well as the intention. Also, a procedure representing 'pip' is started.

Further progression along the time line leads to an activation of the 'pip' plan, to be seen in node 4 in Figure 8. After 'pip' is activated, there is the possibility of a successful termination of 'pip'. It is not possible for 'pip' to abort because of its abort condition.

Advancing further along the time line, a point is reached, where 4 hours have passed since the activation of 'pti' (see node 5 in Fig. 8). From this point on for the next 2 hours, it has to be verified, that a large enough decrease of bilirubin in the blood of the newborn is measured or that the 'pti' plan aborts. This additional proof obligation is represented by node 6 in Figure 8.

Assuming this proof obligation is possible to verify, it is necessary to advance 2 hours further along the time line in the last open premise and close the remaining proof obligations with induction.

5.3 First Error

Proof proceeds as explained in Sect. 5.2 to node 6. There it has to be verified, that the intention is satisfied or the 'pti' plan aborts. However, neither is happening.

abort-condition of pti:

(or (and (bilirubin ≠ phototherapy-intensive) •• NOW
(not (and (bilirubin = phototherapy-normal) •• NOW
(TSB-decrease = no) •• NOW **))**
(and (bilirubin = phototherapy-intensive) •• NOW
(or (and (TSB-decrease = yes) •• ([4h, -] [-, 6h] [-, -] SELF)
(TSB-change < 1) •• ([4h, -] [-, 6h] [-, -] SELF))
(TSB-decrease = no) •• ([4h, -] [-, -] [-, -] SELF))))

If the bilirubin level is not phototherapy-intensive, only the first half of the abort condition can become true. Therefore, with the TSB on a level different than phototherapy-intensive, the plan may only abort, if the bilirubin is not on level phototherapy-normal or **some** decrease in TSB is measured.

The automatically generated proof obligation is now to verify, that TSB level is unequal to `phototherapy-normal` or that there is some decrease of bilirubin measured now. If either did hold, the abort condition would have triggered, resulting in an abortion of the 'pti' plan which would in consequence satisfy

the intention. Verification of this obligation is not feasible, as no side formulas imply either the decrease or the bilirubin blood level. The culprit here is the negation of

(and (bilirubin = phototherapy-normal) •• NOW
(TSB-decrease = no) •• NOW **)**

Without this negation, the plan would react according to the intuition, aborting, once the bilirubin is on level `phototherapy-normal` and **no** decrease is measured. However, simply discarding the negation would lead to the next problem. If a minor decrease is measured, the plan still would not abort, but also the intention would not be satisfied.

corrected abort-condition of pti:

(or (and (bilirubin ≠ phototherapy-intensive) •• NOW
(and (bilirubin = phototherapy-normal) •• NOW
(or (TSB-decrease = no) •• NOW
(TSB-change < 1) •• NOW **)))**
(and (bilirubin = phototherapy-intensive) •• NOW
(or (and (TSB-decrease = yes) •• ([4h, -] [-, 6h] [-, -] SELF)
(TSB-change < 1) •• ([4h, -] [-, 6h] [-, -] SELF))
(TSB-decrease = no) •• ([4h, -] [-, -] [-, -] SELF))))

5.4 Second Error

After modifying the abort condition, the proof has to be reconstructed. Fortunately, most of the original proof can be replayed and the degree of automation is about 95%. The second proof attempt fails at node 6, where again it is necessary to verify that either plan `'pti'` is aborted or the intention is satisfied.

The failed proof attempt can be analyzed to discover a principle problem with time annotations. The original time annotation in the abort condition reads

[4h,_][_,6h][_,_] enter(activated, 'pti')

As the latest finishing shift is given, the time-annotated condition must be finally false (see Sect. 2.3). In order to determine whether the abort condition holds, we have to wait until the condition is falsified. If, after four hours there is a decrease, but the change is not large enough, the abort condition does not necessarily trigger. This violates the intention.

In our example, we must neither give an earliest starting shift nor a latest finishing shift. The corrected time annotation reads:

[_,6h][4h,_][_,_] enter(activated, 'pti')

5.5 Final Verification

With this new time annotation and the changed abort condition, the proof is replayed once more. Again, an automation grade between 95 and 99% is achieved.

It can now be shown in the proof obligation in node 5 in Fig. 8, that a violation of the intention may not occur without the plan being aborted. Using temporal logic induction the proof can be closed.

All in all, the proof had to be replayed twice and both times the abort condition had to be corrected. Thereafter the proof obligation can be verified. The complete proof consists of more than 600 proof steps. The symbolic execution is done almost automatically apart from the generalizations. Most of the manual work has to be done in the first order proof parts.

It would now be necessary to take the verification results and communicate them back to the guideline developers and Asbru modelers to assess the impact of the flaws found as well as the proposed solution.

6 Related Work

In addition to Asbru, a number of languages have been devised for the representation of medical guidelines (see [11] for a comparison of the most outstanding ones). Many of these languages are not formal enough, e.g. they often incorporate many free-text elements without a clear semantics. Exceptions to this are PROforma [12] and Asbru. The formal semantics defined for Asbru makes formal verification of Asbru guidelines possible. This is, to our knowledge, a rather novel approach in the area of guideline representation languages.

Besides KIV, there are a number of powerful theorem provers, e.g. Isabelle [13], STeP [14], and others, which can also be considered for the verification of medical guidelines. We have focused on KIV, because this theorem prover offers the intuitive strategy of symbolic execution with induction to verify temporal properties of concurrent systems.

While interactive verification is convenient for validating the semantics of Asbru and for reasoning about complex details such as time annotations, it is promising to provide automatic methods to effectively apply formal methods in practice. For this, a number of tools can be considered. In [5], we have used the SMV model checker [15] to verify properties of medical guidelines. Alternatives are SPIN [16], Kronos [17], etc. Especially the latter is an alternative to SMV for verifying time annotations. Dealing with time annotations usually implies the necessity for histories ranging over infinite data. As we expect the effort to find abstractions to be somewhat equal to the effort for interactive verification, we concentrate on the latter.

7 Conclusion

Our overall experiences with the jaundice case study suggest that the quality of medical guidelines could be improved by carrying over standard techniques from software engineering. As a matter of fact, we have identified a small number of errors in the formulation of a single plan of a particular Asbru hierarchy. This is a significant finding, given the degree of complexity of guideline languages. Not less important, we have shown how to apply formal methods to a non-standard

application field. The procedure consists of modelling the guideline in Asbru and then verifying properties with an interactive theorem prover. In [5], we also considered model checking of guidelines.

The verification example has shown that symbolic execution is an intuitive strategy to verify guidelines interactively and offers a high degree of automation. Support for the reuse of proofs is essential, because in practice errors are found, the model is changed and proofs have to be reconstructed frequently.

For model checking Asbru guidelines it has been necessary to abstract from time annotations. Therefore, only part of the errors described in Sect. 5 have been found in [5]. Errors concerning the use of time annotations still requires interactive verification. Furthermore, we have found interactive verification very useful to validate the semantics of Asbru.

We have only considered a small sub part of the overall guideline. The selected part has been nontrivial because of time-annotated conditions and the complex semantics involved. The challenge remains to tackle the complete guideline with interactive verification. For this, we are currently working on a modular strategy based on the well known assumption guarantee approach. It would be interesting to also apply other approaches to the verification of medical guidelines. For this, the jaundice guideline could serve as an interesting case study. Other languages besides Asbru should also be considered.

References

1. Seyfang, A., Kosara, R., Miksch, S.: Asbru 7.3 reference manual. Technical report, Vienna University of Technology (2002)
2. Seyfang, A., Miksch, S., Marcos, M.: Combining diagnosis and treatment using Asbru. International Journal of Medical Informatics **68**((1-3)) (2002) 49–57
3. Marcos, M., Roomans, H., ten Teije, A., van Harmelen, F.: Improving medical protocols through formalisation: a case study. In: Proc. of the Session on Formal Methods in Healthcare, 6th International Conference on Integrated Design and Process Technology (IDPT-02). (2002)
4. Balser, M., Duelli, C., Reif, W.: Formal semantics of Asbru – An Overview. In: Proceedings of IDPT 2002, Society for Design and Process Science (2002)
5. Bäumler, S., Balser, M., Dunets, A., Reif, W., Schmitt, J.: Verification of medical guidelines by model checking – a case study. In: SPIN conference proceedings. (2006) to appear.
6. Balser, M., Reif, W., Schellhorn, G., Stenzel, K., Thums, A.: Formal system development with KIV. In Maibaum, T., ed.: Fundamental Approaches to Software Engineering. Number 1783 in LNCS, Springer-Verlag (2000)
7. AAP: American Academy of Pediatrics, Provisional Committee for Quality Improvement and Subcommittee on Hyperbilirubinemia. Practice parameter: management of hyperbilirubinemia in the healthy term newborn. Pediatrics **94** (1994) 558–565
8. Balser, M.: Verifying Concurrent System with Symbolic Execution – Temporal Reasoning is Symbolic Execution with a Little Induction. PhD thesis, University of Augsburg, Augsburg, Germany (2005)

9. Stegers, R.: From natural language to formal proof goal: Structured goal formalisation applied to medical guidelines. Master's thesis, Vrije Universiteit, Amsterdam (2006)
10. Balser, M., Duelli, C., Reif, W., Schellhorn, G.: Verifying concurrent systems with symbolic execution. Journal of Logic and Computation **12**(4) (2002) 549–560
11. Peleg, M., Tu, S., Bury, J., Ciccarese, P., Fox, J., Greenes, R., Hall, R., Johnson, P., Jones, N., Kumar, A., Miksch, S., Quaglini, S., Seyfang, A., Shortliffe, E., Stefanelli, M.: Comparing Computer-interpretable Guideline Models: A Case-study Approach. Journal of the American Medical Informatics Association **10**(1) (2003) 52–68
12. Fox, J., Johns, N., Lyons, C., Rahmanzadeh, A., Thomson, R., Wilson, P.: PROforma: a general technology for clinical decision support systems. Computer Methods and Programs in Biomedicine **54** (1997) 59–67
13. Paulson, L.C.: Isabelle: A Generic Theorem Prover. LNCS 828. Springer (1994)
14. Manna, Z., the STeP group: Step: The stanford temporal prover. Technical report, Computer Science Department, Stanford University (1994)
15. McMillan, K.L.: Symbolic Model Checking. Kluwer Academic Publishers (1990)
16. Holzmann, G.J.: The SPIN Model Checker. Addison-Wesley (2003)
17. Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., Yovine, S.: Kronos: A model-checking tool for real-time systems. In Hu, A.J., Vardi, M.Y., eds.: Proc. 10th International Conference on Computer Aided Verification, Vancouver, Canada. Volume 1427., Springer-Verlag (1998) 546–550